Pedagogical Patterns: their Place in the Genre

Sally Fincher
Computing Laboratory
University of Kent at Canterbury
Canterbury, Kent, UK
+44 1227 824061
S.A.Fincher@ukc.ac.uk

Ian Utting
Computing Laboratory
University of Kent at Canterbury
Canterbury, Kent, UK
+44 1227 823811
I.A.Utting@ukc.ac.uk

ABSTRACT

This paper describes some constituents of patterns and pattern languages and examines the Pedagogical Patterns endeavour against them. Some observations are made with regard to how pattern languages are developed and some suggestions as to how these might be applied to pedagogical patterns are made.

Categories and subject descriptors: K.3.2 [Computers and Education]: Computer and Information Science Education --- Computer Science Education

General Terms: Design

Additional Keywords: Pattern Languages

1. INTRODUCTION

We have previously described the five components ("functional requirements") we believe to be necessary in a Pattern Language, and commented on the extent to which the pedagogical patterns endeavour (at that time) embraced them [1]. As the pedagogical patterns movement has matured, and as Pattern Languages themselves have been explored in other domains, we look again in this paper at how pedagogical patterns are constructed (individually, and especially collectively) and to what uses they may be put.

2. PEDAGOGICAL PATTERNS TODAY

In its original incarnation, the pedagogical pattern collection was just that. A collection of single examples, contributed by many people, on a common theme. Today (November 2001) the thrust of the endeavour has changed, with the emphasis now on smaller, more tightly constrained areas ("How to run a Seminar") either singly authored, or constructed by a small group of people [2]. This approach, more tightly coupled to specific contexts, is more coherent, and therefore successful. However, there remain aspects of pattern languages that these examples do not embody. In the next section we outline what we believe to be the crucial aspects of pattern languages, and indicate where the Pedagogical pattern collections do (or might) incorporate them.

3. WHAT CHARACTERISES PATTERN LANGUAGES?

3.1 Functional Requirements

A Pattern is most usually described as "a solution to a problem in a context". However, we prefer the more specific definition of Dirk Riehle and Heinz Zullighoven [3], "A pattern is the abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts". Patterns sit within a structure, a "language" which relates each one to the collection and the domain. It is relatively easy to capture what might be thought of as the "functional requirements" of a Pattern Language. They are: Capture of Practice, Abstraction, Value System, Structuring Principle and Presentational Form.

3.1.1 Capture of Practice

A Pattern must be about something specific. It is not about an idea, it is not about something that "might be" (or, even worse, something that "should be") it is about things that exist in the world. A pattern must capture examples of things that embody a certain common quality—the examples may be disparate, but they must all exemplify the same essential truth. The nature of this truth or quality is often debated. It has been variously described as "the invariant property common to all [instances] which succeed in solving the problem" or "the quality without a name" or "aliveness". These are concepts which sometimes engender discomfort, but they are fundamental to the pattern endeavour. Here is how this concept is described with respect to buildings:

> We have been taught that there is no objective difference between good buildings and bad, good towns and bad.
>
> The fact is that the difference between a good building and a bad building, between a good town and a bad town, is an objective matter. It is the difference between health and sickness, wholeness and dividedness, self-maintenance and self-destruction. In a world which is healthy, whole, alive and self-maintaining, people themselves can be alive and self-creating. In a world which is unwhole and self-destroying, people cannot be alive: they will inevitably themselves be self-destroying, and miserable.
>
> But it is easy to understand why people believe so firmly that there is no single, solid basis for the difference between good buildings and bad.

It happens because the single central quality which makes the difference cannot be named. [4]

This seems, to us, to be resonant of teaching. Objectively, it is hard to identify the differences between two lectures—both may be in the same lecture theatre, both lecturers speak for the same amount of time, both may use the same kind of visual aids, in the same medium—and yet, from experience, we know that one may be good, whole, alive and engaging; the other, unwhole and self-destructing. Pedagogical patterns, however, do not (very much) capture practice *across* instances. The majority of the patterns appear to be codifications of single pieces of practice, or practice from single practitioners. At the current time, there seems to be little variety of input from which selections of practice, exemplifying the desired qualities, can be selected and captured.

3.1.2 Abstraction

This is a bone of some contention in the wider patterns community (that is, wider than a single domain). There are those who staunchly uphold that "abstraction" is at best a meaningless aim and at worst a dangerous one; that "complex things are complex" and that abstracting away from that complexity makes a false goal of simplification. [5] But, as pedagogues, we know that abstraction is a very difficult step to take [6]; that learners find it difficult to grasp the principles embodied in a single example (or a series of single examples) then isolate it as the common referent they all share (that is, abstract from the details to the principle) and apply that principle in novel situations. In equal fashion, novices appear to have a poor grasp of detail and little appreciation of what is (and is not) important in a given situation. Both of these extremes are addressed in a good pattern, which must abstract to a quality from a set of examples and at a level which is immediately graspable. If it is too abstract, then it will not be apparent; if it is too concrete, then it will not be perceived as separate from the detail of the examples themselves. Finding the correct level of abstraction is a teaching skill; it is also a goal of the best patterns.

3.1.3 Value System

Patterns don't simply represent "any old way" of doing things, but a good way, a better way. At the very least they represent a purposeful way. These values are not explicated separately from the Patterns themselves. Patterns don't justify the values they embody; the values inform the identification of Patterns.

For example, one of Alexander's patterns, number 178 *Composting Toilets* [7], clearly does not have widespread use. However, if the values that inform it—values of environmental concerns for sustainability, were widely held—then instantiation of this pattern would be widespread.

3.1.4 Structuring Principle

Patterns do not exist in isolation; and the links between patterns are as vital as the components themselves. The Structuring principle is what organises patterns into a whole. An excellent example of this can be seen in the anthology of poetry by the UK Poet Laureate, Andrew Motion [8]. In this work, the poems are arranged not alphabetically by author (or title); nor chronologically by when they were written, nor chronologically by when the author lived; nor categorically, by external categories discussed and agreed upon, such as "The Pre-Raphaelites", "The War Poets" or "The Metaphysical Poets". Here, the poems are arranged in a series of ten concentric circles: Self, Home, Town, Work, Land, Love, Travel, War, Belief and Space. This arrangement is a profound embodiment of a structuring principle that everyone can understand, can relate to. We all have meanings for these categories and most of us can find one of more poems that we should like to place within them. But use of this structuring principle carries additional significance: the act of placement of a poem (within, perhaps, *Work* rather than *Self*) speaks to the values of a specific world-view, not a generic one.

By this structure we recognise something else, too. That the placing of a poem within one of these categories is as significant as the choice of poem itself. And that the relationship between the poems within a category (and the relationship of that category to another category) is also meaningful. That the act of placement within this system is not merely one of organisational convenience, of being able to "put your hand on them" when you need them again, as would be the case with an alphabetic organisation.

This sort of structure, which allows the design and construction of whole environments, not just fragments and facets is the "Language" that individual Patterns form.

Structuring principles are surprisingly difficult to identify, and successful ones are intimately connected to domain. (Architectural patterns, for instance, are arranged by scale, from cities to houses to rooms). What an appropriate structuring principle of the domain of pedagogy (or, more tightly, for example, pedagogy in tertiary-level computer science teaching) might be, is hard to construct. The most "structured" of the current pedagogical pattern collections is "Seminars" which takes the teaching sequence—from preparation through to reflection—as its structuring principle.

3.1.5 Presentational Form

The presentation of a Pattern is often mistaken for the thing itself. Patterns do have a distinguished form, and it is often tinkered with, (see [9]) yet certain common elements persist. For the most part, the common elements of the form are definitional: if a pattern is a "solution to a problem within a context" then it is not surprising that Patterns almost universally contain a problem statement and a solution statement. That is about the extent of commonality of expression. In a "pure" Alexandrian form, what comes between these two is an exposition of the reasons why one would make this design choice, drawn from research and some of the many examples of its use. It is preceded by a statement of the "larger" Patterns of which the one in question can be seen as a component, and followed by a list of the "smaller" Patterns which can be used to comprise this. Pedagogical patterns are still striving for a common form, appropriate to the pedagogic domain. The current collection exhibits two differing forms, and the form which was suggested by the project originators [10] is used by only one of the current proponents.

3.2 Non-functional Requirements

As we know, however, a system is never composed solely of functional requirements; and that in terms of both usability and customer satisfaction, it is more often the non-functional requirements that make the difference. In the case of the non-functional requirements of Patterns and Pattern Languages they are not "speed of response" or "ease of use"; however, they do lie in equally intangible areas; we call these areas "non-obvious"; "insight", "generative power" and "communicative power". As is also often the case in systems specification, non-functional requirements are frequently (although not inevitably) related to functional requirements. Where we believe this to be the case, we have indicated such relationships.

3.2.1 Non-obvious (Paired with Capture of Practice)

One of our favourite architectural patterns is 159 *Light on Two Sides of Every Room* [7]. Simply put, this pattern says that people prefer to inhabit rooms that have natural light from two sources, indeed that they gravitate towards such rooms, and so every room in a house should be designed in such a way that this is achievable.

Light on Two Sides of Every Room is not "obvious". *Build a Room with Windows* is obvious. Furthermore, the choice of specific windows: frames, shutters, the size and number of panes, the materials (wood, aluminium, UPVC), single, double or triple glazing are choices dictated by a combination of availability, locality, specific site, the preference and budget of the client and – most importantly – the skill of the designer. The domain of this "lower" level of design choices, these details of implementation, are properly the province of the architect/builder. That is precisely what they are trained for, where their professional expertise resides: they know what sort of decisions can be made, together with their associated constraints and subsequent design trade-offs. Patterns will help you with precisely *none* of these choices.

Light on Two Sides of Every Room is an exploration of a different level of design space. It is not to do with **how** you put windows in a room, or that you do so at all, but with **why** you do so. It is drawn from considerable domain expertise, supplemented with extensive exploration and reading of others' work. Sadly, many of the Pedagogical patterns still capture practice that is obvious (Gold star; Open the Door [2]) that describes our stuff-in-trade, our implementation detail, not the rationale that allows us to make differences in our design of learning experiences.

3.2.2 Insight (Paired with Abstraction)

The design insights that are conveyed by patterns rest on deeper and more complex issues than details of implementation, they distil the expertise that makes the choice of solution appropriate. A pattern provides a solution to a recurring problem, yes. But a design choice is always made for a reason, and patterns provide that reason. As Brad Appleton puts it "a pattern does more than just identify a solution, it also explains *why the solution is needed*". [11]

3.2.3 Generative (Paired with Structuring Principle)

By generative we do not mean that a set of design rules (be they guidelines, rationales, style guides, curriculum templates or handbooks) "automagically" create a complete design. Design is always a creative act. Patterns can be seen to provide a collection of re-useable ideas, but never complete designs

What we *do* mean by "generative" is that a single pattern is, more-or-less, useless. It's the Structuring Principle that allows a user to find an appropriate pattern at an appropriate level when they need one, combined with the driving coherence of the Value System that allows a complete design to be generated which expresses *a certain way of doing things*. [12]

3.2.4 Communicative Power (Paired with Presentational Form)

Pattern Languages have an unusual communicative power. Partly this is because they all have a name, and this becomes used as synecdoche, a shorthand form, to represent the whole of the pattern. For example, in programming terms, to say to someone "You need a Singleton/Factory/Flywheel here" is to convey a complete approach to the solution of a problem in a very compressed manner. Patterns Languages are also powerful at communicating design concepts *across* communities.

Tom Erickson describes an example of this power in his paper *Lingua Francas for Design: Sacred Places and Pattern Languages* [13] in which he relates the work of urban designer Randolph Hester in the town of Manteo in North Carolina. Hester was brought in to work on a plan for achieving economic renewal without sacrificing the town's character. One of the things that he and his team did was to map the "sacred structures"; the places that the residents valued, the places which made Manteo the sort of town they wanted to live in. Interestingly, "these places [were] almost universally unappealing to the trained professional eyes of an architect, historian, real estate developer, or upper-middle-class tourist." and "only two were protected by historic preservation legislation … that is, the existing planning and legal mechanisms that were intended to help preserve the character of places missed most of what the residents of Manteo actually valued". Not only was the list of "sacred structures" a driving force for the re-development at the time, it was still being used and referred to seven years later. As Erickson concludes, "this is an amazing and inspiring result, perhaps the highest goal to which a designer can aspire. Hester's work in Manteo resulted not only in a plan for achieving economic renewal without sacrificing the town's character (the explicit goal he was employed to achieve), but it also resulted in a *shared, self-sustaining system of beliefs and values* that enabled the plan to be realized over a much longer period of time".

This "communicative power" is partly a function of the expression of the value system as it is exposed through the Patterns, but it is also a function of the format in which they are presented. As well as the name, one of the important parts of a pattern is the "sensitising example", a representation of the *use* of the pattern, which sensitises the reader to the wholeness of the solution. In Alexander's work this function is fulfilled by photographs of places; in GoF, code fragments. As yet, there is nothing that fulfils this for pedagogical patterns. (Although Jutta Eckstein [14] does preface her patterns with a quotation, the function of this is unclear).

4. THE WAY FORWARD?

In current form, Pedagogical patterns still lack widespread acceptance.

Partly we believe this is because they miss some of the requirements: they are either so abstracted from the domain (of tertiary computer science education), and therefore generic, that they lack insight; or they are so tightly coupled to specific instances of practice that they are not transferable. The chosen form(s) lack some of the elements that provide patterns with their peculiar communicative power; sometimes they capture practice which is obvious, sometimes the lack of a value system it is difficult to generate new designs from the solutions they propose.

Partly, we suggest that this is because of the ways in which they are being generated, and the material they are drawing on.

The processes of creating a successful pattern language are rarely modelled, or talked of. Consequently, notions of how to proceed successfully are only found in fragments of description from pattern language authors. However, there are some fragments. The best-known pattern languages are still *A Pattern Language* [7] and *Design Patterns: Elements of Reusable Object-Oriented Software* [15]*;* and they were created in similar ways. Both involved a small group of like-minded people (that is people who shared, or forged, a common value system). Both involved conscious effort over a considerable period of time, drawing on an immense background of joint domain expertise, empirical studies, theoretical and bibliographical knowledge and other sources of primary (and secondary) data. Alexander says little of this, but does comment that it couldn't have been achieved without photocopiers; Vlissides [16] says that it took four years to generate the 23 GoF patterns, and that much "archaeology" was involved in identifying and characterising recurrences, and that each pattern was then iterated and re-written 10-20 times. Others reinforce this. In one of the most complete descriptions of pattern language creation, Brown and Whitenack [17] say "To discover the patterns we first immersed ourselves in the literature and subject area. We found our patterns in numerous places … our own experience … studying the documentation of existing frameworks … reading the OO literature … feedback from our colleagues".

In our own experience [18] in a project analogous to the creation of a pattern language, a group of people who were all experienced in a technique (using projectwork to teach Computer Science) came together to explore the use of projectwork within the discipline. We first undertook a large survey of this specific technique only; we swamped ourselves with detail and example. Over three years our small group debated (and argued) the underlying issues. We did not all agree with each other, but we did come to intimately understand why and where we differed, and to share a common idea of what was important. Only at the very end of this process were we able to distil, iterate and refine our examples; this would have been impossible at the start of the project. It would have been extremely difficult for a less closely connected group to have accomplished this, and practically impossible for any one person alone.

The pedagogical pattern leaders may be the necessary sort of small, dedicated group, that a patterns endeavour needs (although there is currently little evidence of a shared value system). However, their major obstacle may lie in gathering the necessary type and *quantity* of examples. For, in pedagogy, this crucially means taking ourselves outside of our own classrooms. Unlike architecture, or software or interaction design, there are no public, apprehensible artefacts to draw on. We can, maybe, rely on the pedagogical pattern leaders to do this for us: to read, talk, endlessly enquire and sift the literature for examples.

Given the problematic nature of examples in this field, however, perhaps a different, more distributed model might be effective. For example, if the pedagogical patterns leaders were to create a framework, then smaller groups could work within it to reflect on those aspects of practice which particularly interest them; in this way, the inherent problems of privacy and variety of example might be overcome. There has never been a successful example of a pattern language being generated in such a way, but that is not to say it cannot be done.

5. REFERENCES

1. Fincher, S., *Analysis of Design: an exploration of patterns and pattern languages for pedagogy.* Journal of Computers in Mathematics and Science Teaching: Special Issue CS-ED Research, 1999. **18**(3): p. 331-348.

2. *The Pedagogical Patterns Project*. 2001. Available: http://www.pedagogicalpatterns.org

3. Riehle, D. and H. Zullighoven, *Understanding and Using Patterns in Software Development*. Theory and Practice of Object Systems, 1996. **2**(1): p. 3-13.

4. Alexander, C., *The Timeless Way of Building*. 1979, Oxford: Oxford University Press.

5. Gabriel, R., *Patterns of Software: Tales from the Software Community*. 1996, New York: Oxford University Press.

6. Bloom, B. and D. Krathwohl, *Taxonomy of Educational Objectives: The Classification of Educational Goals.* 1956, New York: Longmans, Green.

7. Alexander, C., S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Constructions.* 1977, New York: Oxford University Press.

8. Motion, A., *Here to Eternity*. 2001: Faber & Faber.

9. Fincher, S., *The Pattern Gallery*. 2000. Available: http://www.cs.ukc.ac.uk/people/staff/saf/patterns/gallery.html.

10. PPTOT, *Pedagogic Patterns: Successes in Teaching Object Technology*. 1998. Available: http://www-lifia.info.unlp.edu.ar/ppp

11. Appleton, B., *Patterns and Software: Essential Concepts and Terminology*. Available: http://www.enteract.com/~bradapp/docs/patterns-intro.html.

12. Bransford, J., A. Brown, and R. Cocking, eds. *How People Learn: Brain, Mind, Experience and School (expanded edition)*. 2000, National Academy Press: Washington DC.

13. Erickson, T. *Lingua Francas for Design: Sacred Places and Pattern Languages*. in *DIS 2000*. 2000. Brooklyn, NY: ACM Press.

14. Eckstein, J., *Learning to Teach and Learning to Learn: Running a Course*. 2000. Available: http://www.pedagogicalpatterns.org/examples/LearningAndTeaching.pdf.

15. Gamma, E., et al., *Design Patterns: Elements of Reusable Object-Oriented Software.* 1994, Reading, Massachusets, US: Addison-Wesley.

16. Vlissides, J., *Patterns: The Top Ten Misconceptions.* 1998. Available: http://www.research.ibm.com/designpatterns/pubs/top10misc.pdf

17. Brown, K. and B. Whitenack, *A Pattern Language for Relational Databases and Smalltalk*. 1996. Available: http://www.ksc.com/article2.htm.

18. Fincher, S., M. Petre, and M. Clark, eds. *Computer Science Project Work: Principles and Pragmatics.* 2001, Springer-Verlag: London.